4. The method of claim 1, further comprising:

creating a notify window when a document window is created, wherein the notify window is a child window of the document window; and

receiving by the notify window a message from the operating system when the parent document window is activated by either the application program or by action of a user.

5. The method of claim 1, further comprising, upon receipt of a document window destroy message by a document's notify window, checking for the existence of a child notify window associated with a file that was opened in association with said document, and acting upon said associated file before destroying said document window.

6. The method of claim 5, comprising a user of an application specifying one or more rules to the application, each said rule indicating an action to be taken on said associated file before destroying said document window.

7. The method of claim 6, wherein an action includes encrypting said associated file, wherein encrypting comprises the step of:

creating a new file containing a template of the same format as said associated file, said new file including a header, wherein said herder is not encrypted;

copying to the new file a document summary in readable form;

copying to the new file visual indicia representing that the file is encrypted;

copying encrypted data from the associated file to the new file as a named stream, said encrypted data having a beginning and a length; and

writing a code to a substitute stream to prevent the new file from being written over by a user.

8. The method of claim 7, further comprising the step of including in the header a flag indicative of the type of encryption used on the associated file.

9. The method of claim 7, further comprising the step of including in the header a flag indicative of a handling procedure to be performed on said encrypted file.

10. The method of claim 7, further comprising the step of loading data from the encrypted file into a memory block containing other random data, said memory block having a beginning, wherein an offset from the beginning of the memory block of the data from the encrypted file changes each time the message monitoring program executable code is loaded.

11. The method of claim 7, further comprising the step of including in the header a flag indicative of the length of the encrypted data, and a flag that allows the message monitoring program to search for the beginning of the encrypted data.

12. The method of claim 5, wherein an action includes deleting a temporary file from a mass storage device, wherein the deletion includes the steps of wiping the device sectors of the data contained therein, and renaming the temporary file to a name consisting of one letter prior to deletion.

13. the method of claim 12, wherein the wiping is performed in conformance with the standards set for in the United States National Industrial Security Program Operating Manual

14. The method of claim 6, wherein an action includes compressing said associated file.

15. The method of claim 6, wherein an action includes translating said associated file into another language.

16. The method of claim 6, wherein an action includes converting said associated file into another file format.

17. The method of claim 6, wherein an action includes searching for a virus in said associated file.

18. The method of claim 6, wherein an action includes altering text in said associated file.

19. The method of claim 6, wherein an action includes changing or adding an optional setting to said associated file.

20. The method of claim 6, wherein an action includes spell checking said associated file.

21. The method of claim 6, wherein an action includes creating a backup of said associated file.

22. The method of claim 6, wherein an action includes versioning said associated file under a different name or stream each time said associated file is saved.

23. The method of claim 6, wherein an action includes storing said associated file content in a remote location, and storing a reference of said associated file locally.

24. The method of claim 6, wherein an action includes grammar checking said associated file.

25. The method of claim 1, further comprising monitoring by the message monitoring program window function the creation of a document child window.

26. A method for intercepting a software call to a function contained in a dynamically linked library comprising the steps of:

obtaining from an operating system by an intercepting program the address of an executable program;

locating in a header of the executable program a list of libraries and functions called by the executable program;

substituting a function reference in an import table of the executable program with a reference to a function in a library of the intercepting program; and

storing said function reference in an internal memory structure.

27. The method of claim 26, wherein said executable program header includes an import table, further comprising examining the import table in order to locate the list of libraries and functions called by the executable program.

28. The method of claim 26, wherein the function reference is an ordinal number.

29. The method of claim 26, wherein the function reference is a name.

30. The method of claim 26, further comprising

obtaining the address of a library, said library including an import table; and

substituting a function reference in the import table of the library with a reference to a function in a library of the intercepting program.

31. The method of claim 30, wherein the library address is obtained from the operating system.

32. The method of claim 30, wherein the library address is obtained from the import table of the executable.

33. The method of claim 26, further comprising replacing the address of an API function in the import table of the executable with the address of a function of the intercepting program, said function having a return value comprising a table of addresses of one or more functions, wherein the intercepting program function inspects the table of addresses returned by the API function and inserts one or more substitute function addresses into the table of addresses.